[不太真实的战斗模拟器]设计文档

1. 游戏介绍

内容：《Not Quite Accurate Battle Simulator》（《不太真实的战斗模拟器》）是一款基于 unity3d 技术自主创意开发的 3D 策略类单机游戏。

游戏设计 8 关，其中前 7 关为核心关卡，每一关都有不同的地图和新的玩法，第 8 关是自定义关卡，给玩家提供了自制关卡的空间，丰富了游戏的可玩性。

游戏的策略性基于地图的复杂性，自动对战机制的不确定性和不同单位之间的的数值均衡。

2. 游戏设计

内容：游戏界面友好，由虚拟角色进行引导。通过鼠标部署单位，键盘移动视角，交互合理。图标布局合理清晰。

2.1 界面设计

内容：游戏的主要界面和功能，可以直接截取游戏主要界面，以图文形式加以说明。

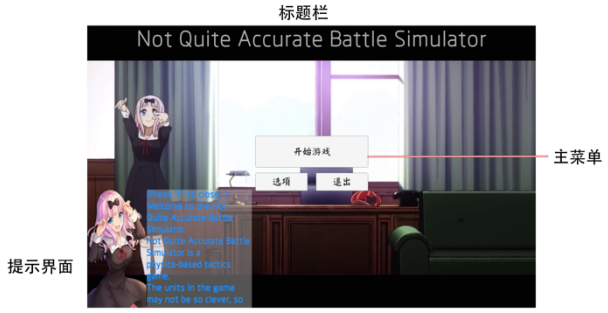

图 1：主菜单

图 2：选关界面

图 3:关卡提示界面

图 4:部署界面

图 5:暂停界面

图 6:战斗界面

图 7:结算界面

2.2 交互设计

内容：说明游戏中玩家的交互方式与按键功能。

鼠标左键：菜单、选择与放置单位、开始战斗。

WASD：控制镜头前左后右位移。

方向键：控制镜头上下左右旋转。

双 Shift：单背/双倍加速。

U：设置放置的是我方单位

I：设置放置的是敌方单位

ESC：暂停。

2.3 流程设计

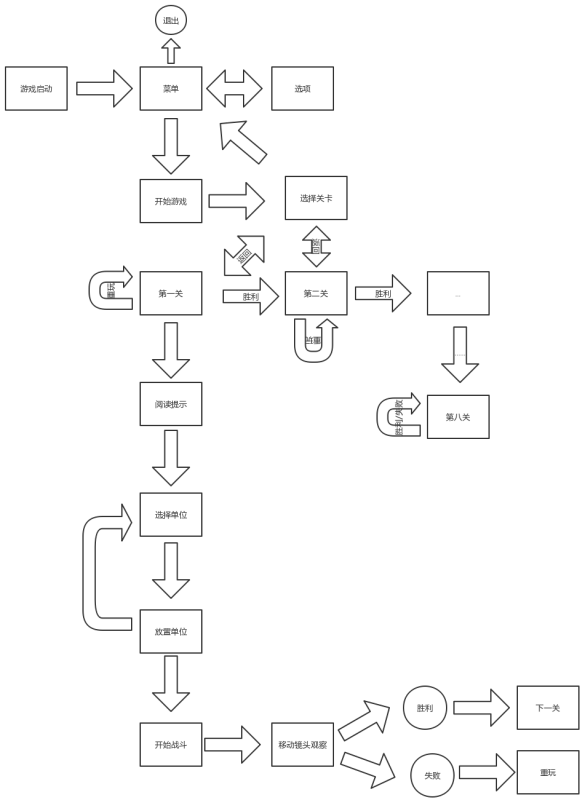

图 8：玩家流程设计

3. 游戏实现

使用 Unity 提供的物理引擎，调用 Start()和 Update()函数逐帧处理游戏内容。

3.1 代码流程

内容：描述游戏程序的主要模块及其功能、游戏的运行流程、模块之间调用关系。

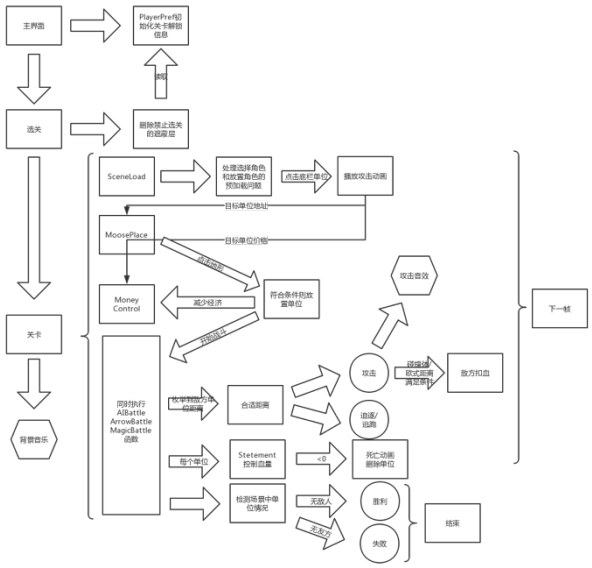

图 9：代码流程

3.2 核心模块

攻击实现：使用刚体（+人物外骨骼）+动画+胶囊碰撞体的方式来触发扣血函数。

寻路实现：使用 NavmeshAgent 模块，预先调整好烘焙参数，河流用圆柱体替代阻止路径出现。

图 10：河流处理

AI 决策：逐帧扫描全体敌方单位，找到最近单位判断距离，设计四档：远、中、近、过近。对不同的角色使用不同的决策手段。比如骑兵在远距离会快速追逐敌人，而魔法师会随着敌人与自己的接近而不断加快逃跑的速度。

同时，所有单位的攻击也主要依赖于查找最近的友方或敌人。

核心代码：

```
private void FindPlayer()
{
    EnemyArray = GameObject.FindGameObjectsWithTag(TargetTag);
    Nearest = null;
    MDistance = findDistance;
    float CDistance;
    for (int i = 0; i < EnemyArray.Length; i++)
    {
        CDistance = Vector3.Distance(this.transform.position, EnemyArray[i].transform.position);
        if (CDistance < MDistance)
        {
            MDistance = CDistance;
            Nearest = EnemyArray[i];
        }
    }
    if (Nearest != null)
    {
        target = Nearest.transform;
        ai.enabled = true;
    }
    else
    {
        ai.enabled = false;
        ani.SetBool("isRun", false);
        ani.SetBool("isWalk", false);
        ani.SetBool("isAttack", false);
    }
}
```

魔法释放：放置相关粒子，并播放特效，同时对周围一定范围内单位进行影响。相关粒子效果自带延时自毁函数。

核心代码：

```
CDtime = CD;
if (type == 1 || type == 2)
{
    hello = (ParticleSystem)Instantiate(Bomb, target.position, Quaternion.identity);
    hello.GetComponent<Destroy>().flag = true;
    hello.Play();
    ani.SetBool("isAttack", true);
    for (int i = 0; i < EnemyArray.Length; i++)
    {
        float Distance = Vector3.Distance(hello.transform.position, EnemyArray[i].transform.position);
        if (Distance < Range)
        {
            if (type == 1) EnemyArray[i].GetComponentInChildren<HPBar>().HealthPoint -= 5;
            else if (type == 2)
            {
                EnemyArray[i].GetComponentInChildren<HPBar>().HealthPoint -= 0.5f;
                if (EnemyArray[i].GetComponent<Statement>().speedmult >= 0.1f) EnemyArray[i].GetComponent<Statement>().speedmult += 0.5f;
            }
        }
    }
}
else if (type == 3)
{
    if (TargetTag == "Player") FriendArray = GameObject.FindGameObjectsWithTag("Enemy");
    else if (TargetTag == "Enemy") FriendArray = GameObject.FindGameObjectsWithTag("Player");
    Nearest = null;
    MDistance = 0.0f;
    float CDistance;
    for (int i = 0; i < FriendArray.Length; i++)
    {
        CDistance = Vector3.Distance(this.transform.position, FriendArray[i].transform.position);
        if (CDistance > MDistance)
        {
            MDistance = CDistance;
            Nearest = FriendArray[i];
        }
    }
    if (Nearest != null)
    {
        target = Nearest.transform;
        hello = (ParticleSystem)Instantiate(Bomb, target.position, Quaternion.identity);
        hello.GetComponent<Destroy>().flag = true;
        hello.Play();
        ani.SetBool("isAttack", true);
        for (int i = 0; i < FriendArray.Length; i++)
        {
            float Distance = Vector3.Distance(hello.transform.position, FriendArray[i].transform.position);
            if (Distance < Range)
            {
                FriendArray[i].GetComponentInChildren<HPBar>().HealthPoint += FriendArray[i].GetComponentInChildren<HPBar>().MaxHealthPoint*0.5f;
            }
        }
```

4. 参考资料

[1] Unity 官方文档 https://docs.unity3d.com

[2] Google 到的一大堆相关教程 https://google.com